A Deep Dive into BianLian Ransomware

Prepared by: Vlad Pasca, Senior Malware & Threat Analyst

SecurityScorecard.com
info@securityscorecard.com

Tower 49
12 E 49th Street
Suite 15-001
New York, NY 10017
1.800.682.1707

Table of contents

Executive summary

BianLian ransomware is a Golang malware that performed targeted attacks across multiple industries in 2022. The ransomware employed anti-analysis techniques consisting of API calls that would likely crash some sandboxes/automated analysis systems. The malware targets all drives identified on the machine and deletes itself after the encryption is complete.

The files are encrypted using the AES256 algorithm (Golang package AES), and as opposed to other ransomware families, the AES key is not encrypted by a public key and is not stored in the encrypted files. We believe that decryption is possible by recovering the ransomware encryptor using forensics tools. The extension of the encrypted files is changed to ".bianlian."

Analysis and findings

SHA256: eaf5e26c5e73f3db82cd07ea45e4d244ccb3ec3397ab5263a1a74add7bbcb6e2

The malware is a 64-bit executable compiled with Golang. The Build ID shown in figure 1 is a unique representation of the file and its content. Also, the path shown below contains the "crypt28" string:

File pos	Mem pos	ID	Text
A 000000000601	00000000058D	0	Go build ID: "H40nAXi0HAA8phzv9-cb/qCmr9jSfyS54gBjEKYHI/3NP6oNV505RosziU-nxb/ldC38qRUGilCycasAQgK"
A 000000148C29	000000148BB5	0	build-gcflags=all=-trimpath=/home/jack/Projects/project1/crypt28
A 00000166A5D	0000001669E9	0	/home/jack/Projects/project1/common/crypt.go
A 00000166A8A	00000166A16	0	/home/jack/Projects/project1/common/helpers.go
A 00000166AB9	00000166A45	0	/home/jack/Projects/project1/common/scanFS_windows.go
A 00000166AEF	00000166A7B	0	crypt.go
A 00000166AF8	00000166A84	0	main.go
A 00000166B00	00000166A8C	0	/home/jack/Projects/project1/common/scanFS.go

Figure 1

The LoadLibraryA API is utilized to load the "kernel32.dll" module into the address space of the process:

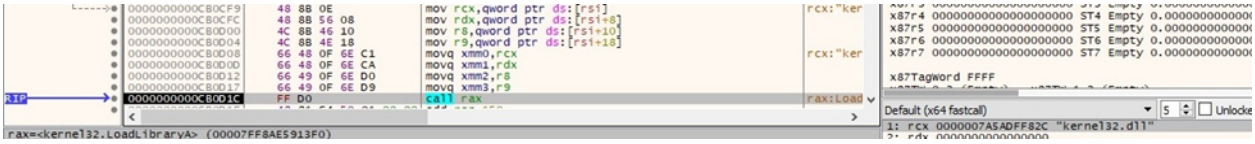

Figure 2

The ransomware retrieves the address of multiple export functions: "AddDllDirectory", "AddVectoredContinueHandler", "LoadLibraryExA", and "LoadLibraryExW" (see figure 3).

Figure 3

It obtains the path of the System32 directory via a function call to GetSystemDirectoryA:

Figure 4

LoadLibraryExA is used to load the following DLLs into the process memory: "advapi32.dll", "ntdll.dll", "winmm.dll", and "ws2_32.dll" (0x800 = **LOAD_LIBRARY_SEARCH_SYSTEM32**):

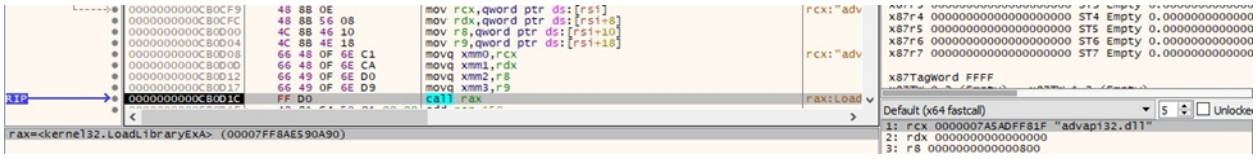

Figure 5

The malicious binary forces the system not to display the Windows Error Reporting dialog using SetErrorMode (0x2 = **SEM_NOGPFAULTERRORBOX**):

Figure 6

The executable registers a vectored exception handler by calling the RtlAddVectoredExceptionHandler function:

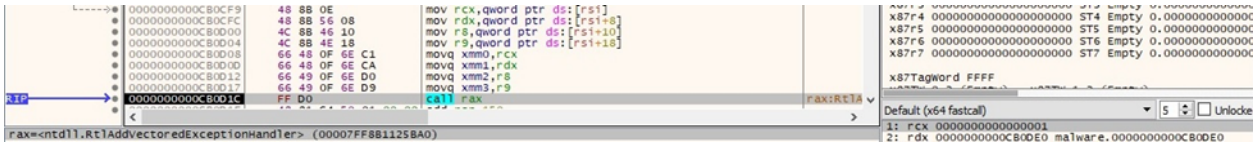

Figure 7

BianLian ransomware creates an unnamed timer object using the CreateWaitableTimerExW routine (0x2 = **CREATE_WAITABLE_TIMER_HIGH_RESOLUTION**, 0x100003 = **SYNCHRONIZE** | **TIMER_MODIFY_STATE** | **TIMER_QUERY_STATE**):

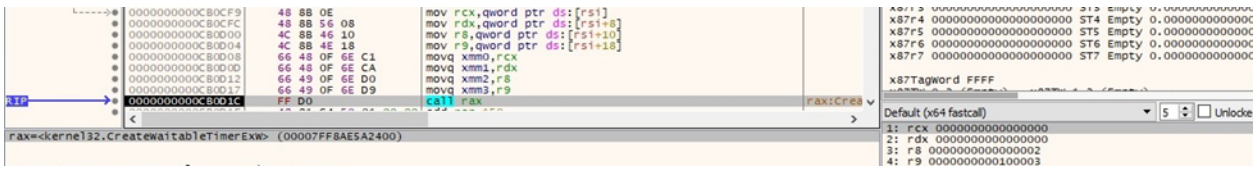

Figure 8

The timeBeginPeriod function is utilized to request a minimum resolution for periodic timers:

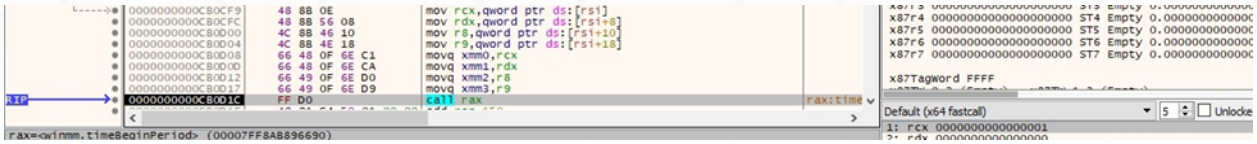

Figure 9

The RtlGetNtVersionNumbers low-level API is used to extract the Windows version numbers:

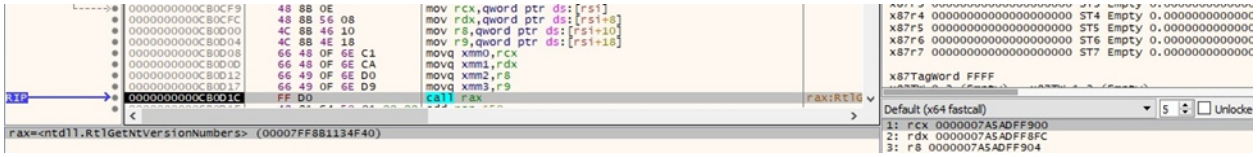

Figure 10

The malware obtains the PEB's (Process Environment Block) address of the current process using RtlGetCurrentPeb:

Figure 11

The ransomware generates 32 pseudo-random bytes via a function call to RtlGenRandom:

Figure 12

It calls the CreateFileA function with a file name consisting of the above bytes and many "A" characters (figure 13). The function call returns a "NAME INVALID" error, and we believe that the threat actor wanted to avoid automated systems/sandboxes using most of the API calls presented so far.

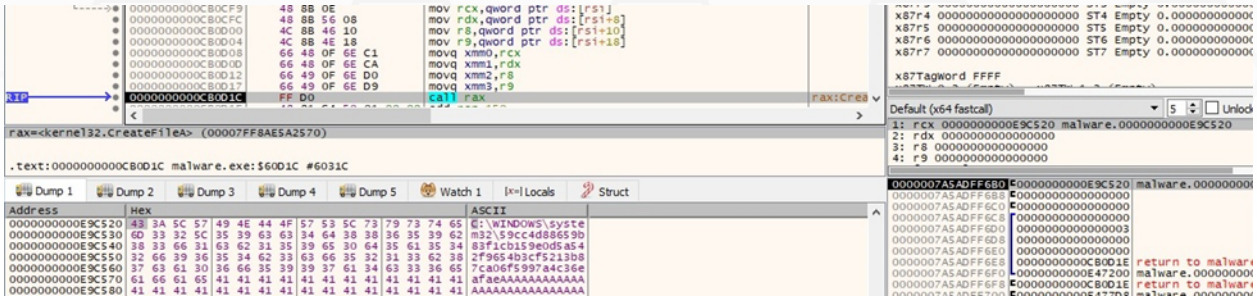

Figure 13

The binary retrieves the affinity mask for the current process and the system:

Figure 14

GetSystemInfo is used to extract information about the current system, as shown in figure 15.

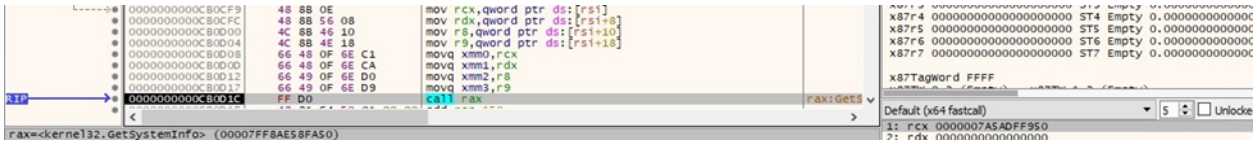

Figure 15

The malicious executable disables dynamic boosting for the current process:

Figure 16

The VirtualAlloc API is used to allocate memory in the address space of the current process (0x3000 = **MEM_COMMIT** | **MEM_RESERVE**, 0x4 = **PAGE_READWRITE**):

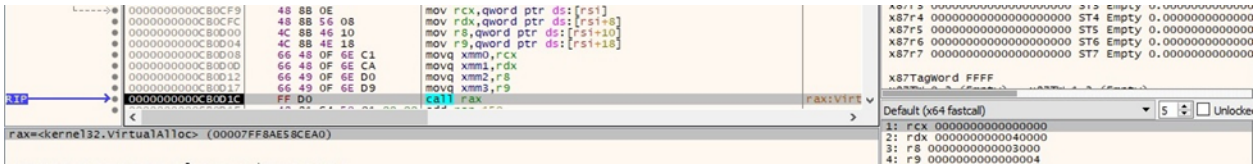

Figure 17

BianLian ransomware retrieves the environment variables for the process using

GetEnvironmentStringsW:

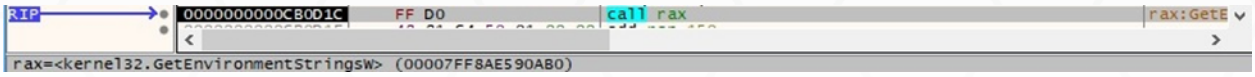

Figure 18

The process adds a HandlerRoutine function to the list of handler functions by calling the SetConsoleCtrlHandler routine (see figure 19).

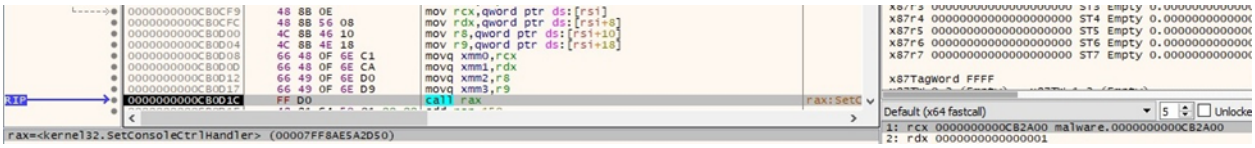

Figure 19

The PowerRegisterSuspendResumeNotification function is utilized to receive notifications when the system is suspended or resumed (0x2 = **DEVICE_NOTIFY_CALLBACK**):

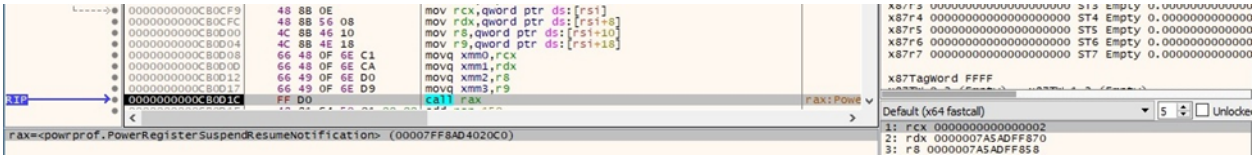

Figure 20

The malware duplicates the current process handle using the DuplicateHandle API:

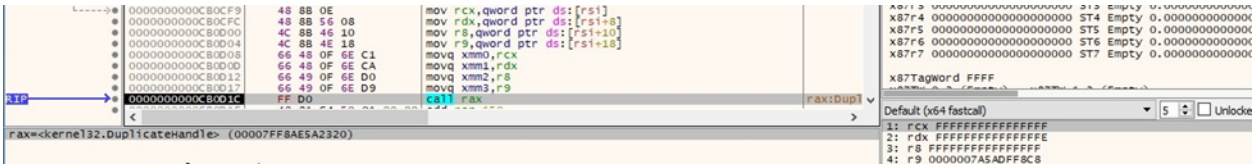

Figure 21

Multiple threads that run the same function (sub_CB0FC0) and are responsible for files' encryption are created:

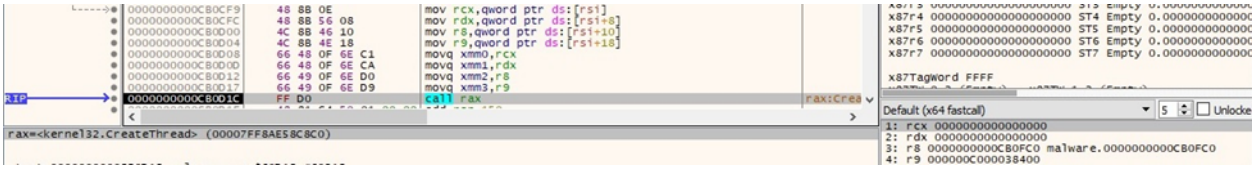

Figure 22

The encryption threads are synchronized using unnamed event objects:

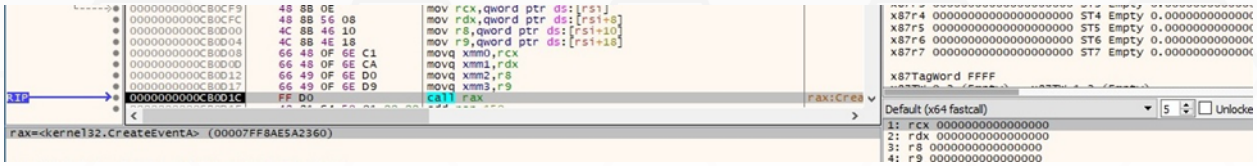

Figure 23

The process sets the event objects to the signaled state using the SetEvent function:

Figure 24

The GetStdHandle routine is utilized to obtain a handle for the standard input device (0xFFFFFFF6 = **STD_INPUT_HANDLE**):

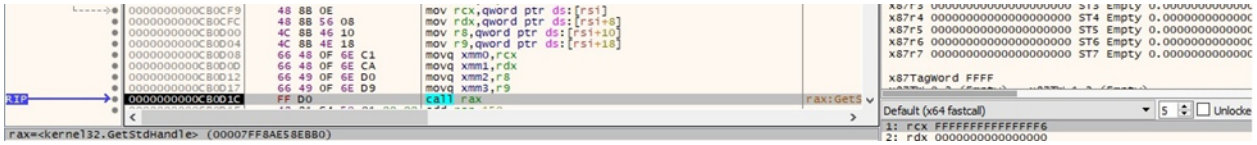

Figure 25

The ransomware initiates the use of the Winsock DLL via a function call to WSAStartup:

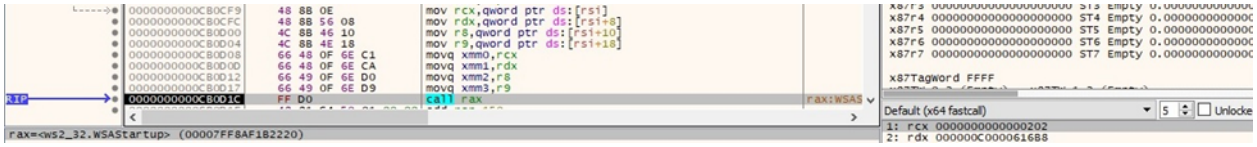

Figure 26

The binary obtains information about the available protocols using WSAEnumProtocolsW:

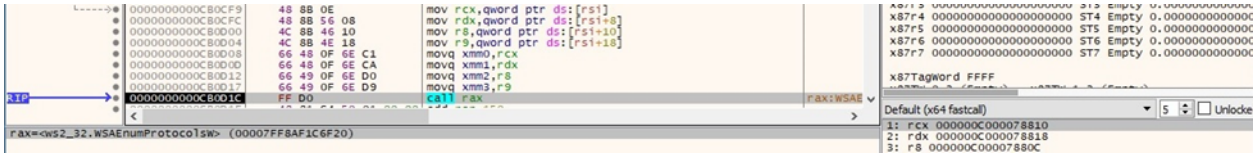

Figure 27

It retrieves the command-line string for the process by calling the GetCommandLineW API:

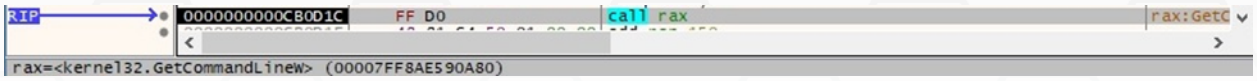

Figure 28

GetEnvironmentVariableW is used to extract the content of the "GODEBUG" environment variable, as highlighted below:

Figure 29

BianLian ransomware calls the GetDriveTypeW function with arguments ranging from "A:" to "Z:" drives:

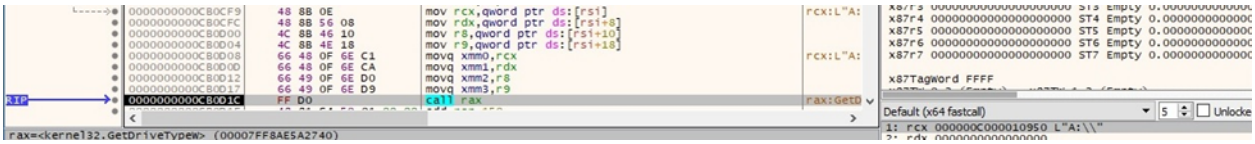

Figure 30

For each identified drive, the process opens it in reading mode using CreateFileW (0x80000000 = **GENERIC_READ**, 0x3 = **FILE_SHARE_READ** | **FILE_SHARE_WRITE**):

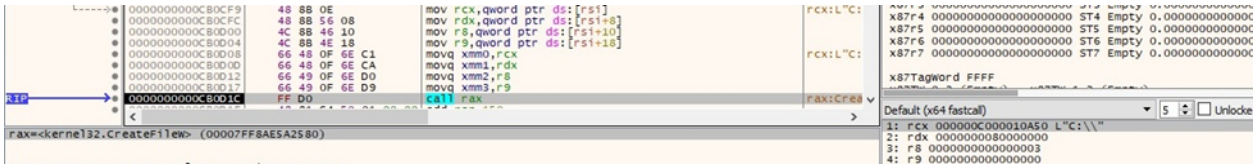

Figure 31

The files are enumerated by calling the FindFirstFileW and FindNextFileW APIs. The ransomware doesn't encrypt executables, drivers, and text files because this would leave the system inoperable.

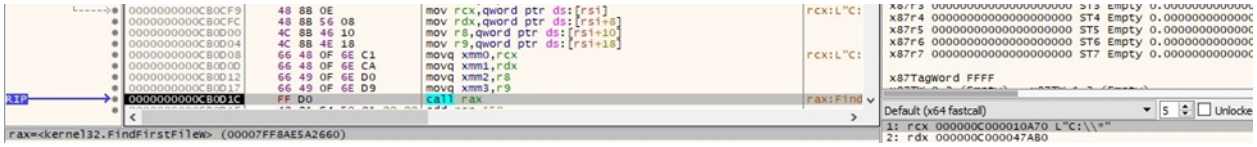

Figure 32

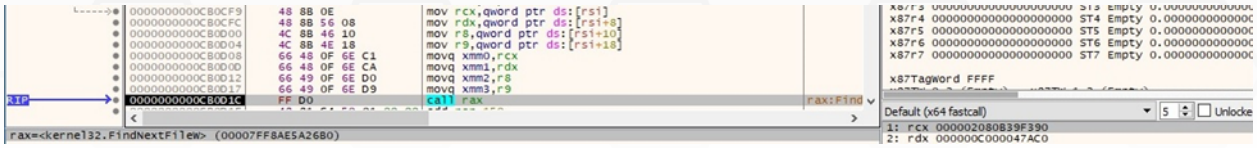

Figure 33

The malware creates a ransom note called “Look at this instruction.txt” in every traversed directory (0x40000000 = **GENERIC_WRITE**, 0x3 = **FILE_SHARE_READ** | **FILE_SHARE_WRITE**):

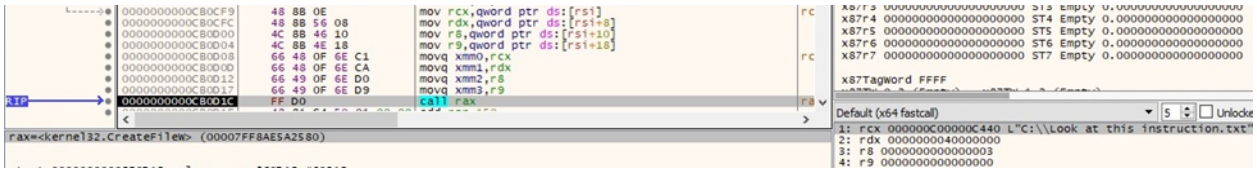

Figure 34

The WriteFile routine is used to populate the ransom note, which contains a hard-coded victim ID, an email address that can be used to contact the threat actor, and the DarkWeb link:

Figure 35

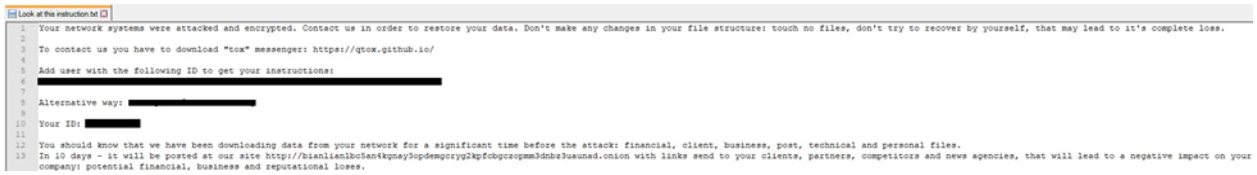

Figure 36

The binary retrieves the path of the executable file of the current process using GetModuleFileNameW:

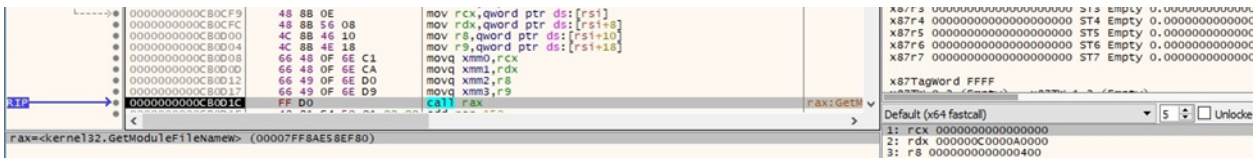

Figure 37

The ransomware extracts the “PATHEXT” environment variable that contains a list of extensions corresponding to executable files (see figure 38).

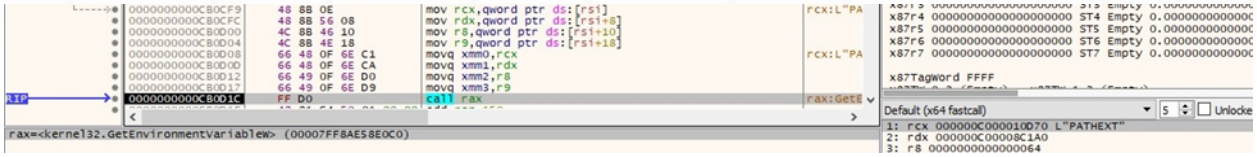

Figure 38

The process is looking for the “cmd.exe” file. It obtains attributes for this file by calling the GetFileAttributesExW function (0x0 = **GetFileExInfoStandard**):

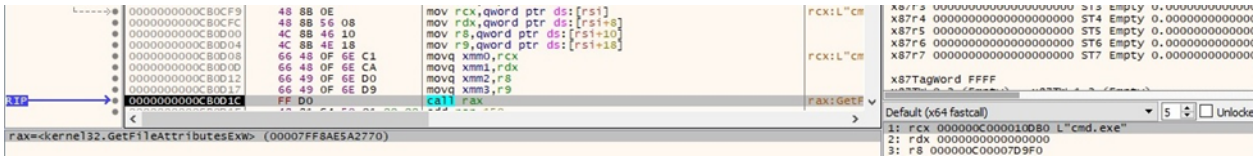

Figure 39

The CreateFileW API is used to confirm the location of the “cmd.exe” executable:

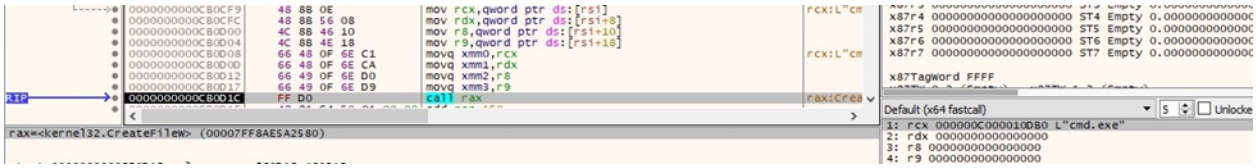

Figure 40

Whether the “cmd.exe” file is not found in the current directory, the malware retrieves the “path” environment variable and concatenates the extracted paths with “cmd.exe”:

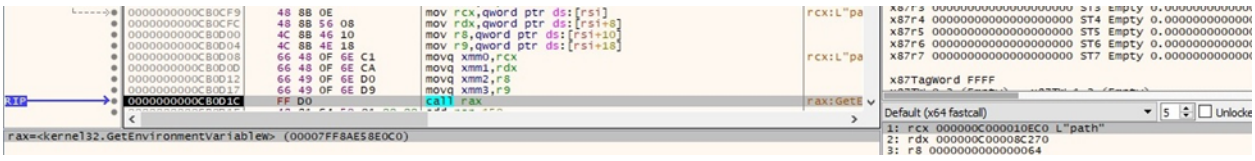

Figure 41

The malicious executable obtains a pseudo handle for the current process via a call to GetCurrentProcess:

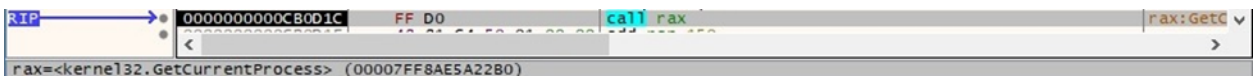

Figure 42

After the encryption finishes, the malware deletes itself:

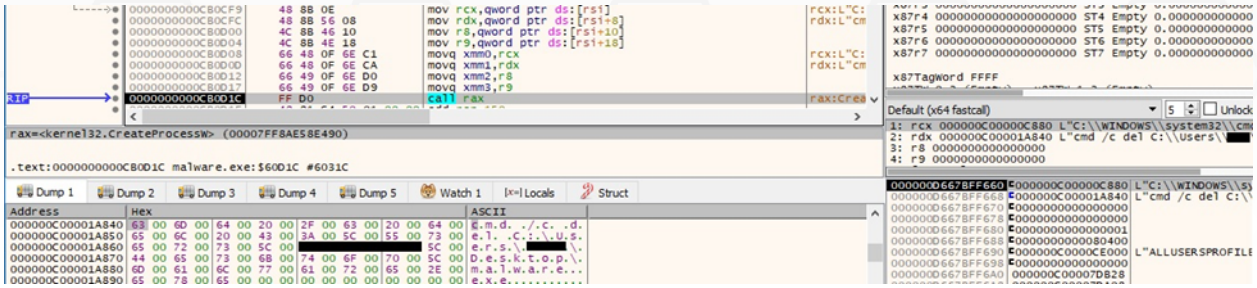

Figure 43

Thread activity – sub_CB0FC0 function

The malware reads the file content using the ReadFile API:

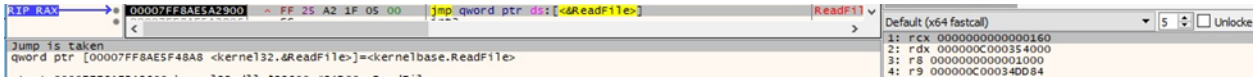

Figure 44

The GetFileType API is utilized to retrieve the file type:

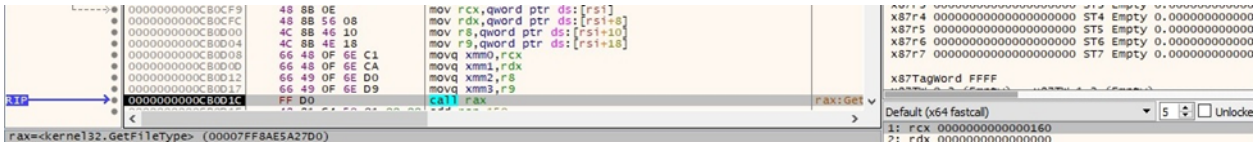

Figure 45

The binary moves the file pointer to the beginning of the file via a function call to SetFilePointerEx (0x0 = **FILE_BEGIN**):

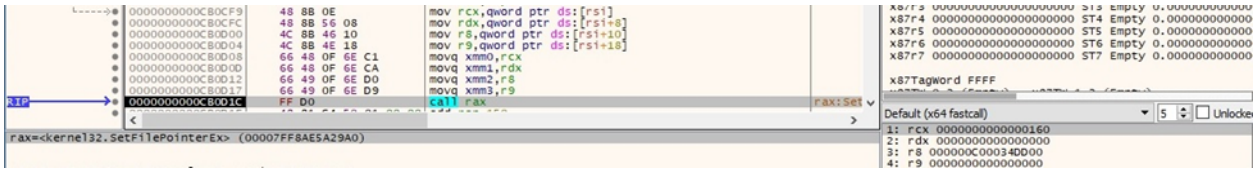

Figure 46

The encryption key is generated based on 32 bytes hard-coded in the ransomware. The “aeskeygenassist” instruction is used to compute 240 bytes, as highlighted below:

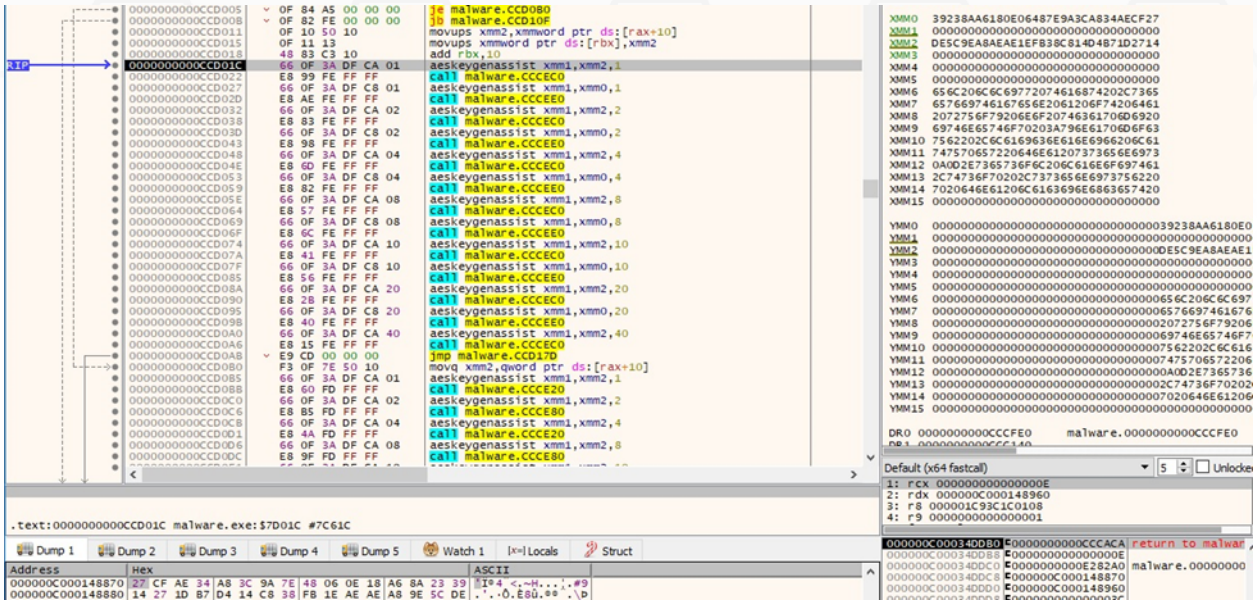

Figure 47

Address	Hex	ASCII
000000C000148870	27 CF AE 34 A8 3C 9A 7E 48 06 0E 18 A6 8A 23 39	'Ï®4¨<.~H...¦.#9
000000C000148880	14 27 1D B7 D4 14 C8 38 FB 1E AE AE A8 9E 5C DE	.'.·Ô.È8û.®®¨.\Þ
000000C000148890	2D 85 B3 F6 85 B9 29 88 CD BF 27 90 6B 35 04 A9	-.³ö.¹).Í¿'.k5.©
000000C0001488A0	6B B1 EF 64 BF A5 27 5C 44 BB 89 F2 EC 25 D5 2C	k±ïd¿¥'\D».ò‰Õ,
000000C0001488B0	10 86 C2 38 95 3F EB B0 58 80 CC 20 33 B5 C8 89	..Â8.?ë°X.Ì 3µÈ.
000000C0001488C0	A8 64 07 C3 17 C1 20 9F 53 7A A9 6D BF 5F 7C 41	¨d.Ã.Á .Sz©m¿_|A
000000C0001488D0	DB 96 41 30 4E A9 AA 80 16 29 66 A0 25 9C AE 29	Û.A0N©ª..)f %.®)
000000C0001488E0	97 BA E3 66 80 7B C3 F9 D3 01 6A 94 6C 5E 16 D5	.ºãf.{Ãùó.j.l^.Õ
000000C0001488F0	8B D1 42 60 C5 78 E8 E0 D3 51 8E 40 F6 CD 20 69	.ÑB`Åxèàó Q.@öÍ i
000000C000148900	D5 07 54 9F 55 7C 97 66 86 7D FD F2 EA 23 EB 27	Õ.T.U|.f.}ýòê#ë'
000000C000148910	BD 38 8E E7 78 40 66 07 AB 11 E8 47 5D DC C8 2E	½8.çx@f.«.èG]ÜÈ.
000000C000148920	99 81 BC AE CC FD 2B C8 4A 80 D6 3A A0 A3 3D 1D	..¼®Ìý+ÈJ.Ö: £=.
000000C000148930	97 1F 2A 07 EF 5F 4C 00 44 4E A4 47 19 92 6C 69	..*.ï_L.DN¤G..li
000000C000148940	4D CE EC 57 81 33 C7 9F CB B3 11 A5 6B 10 2C B8	MÎìW.3Ç.Ë³.¥k.,¸
000000C000148950	1D 6E 46 78 F2 31 0A 78 B6 7F AE 3F AF ED C2 56	.nFxò1.x¶.®?¯íÂV

Figure 48

The process performs the AES InvMixColumn transformation using the “aesimc” instruction (see figure 49).

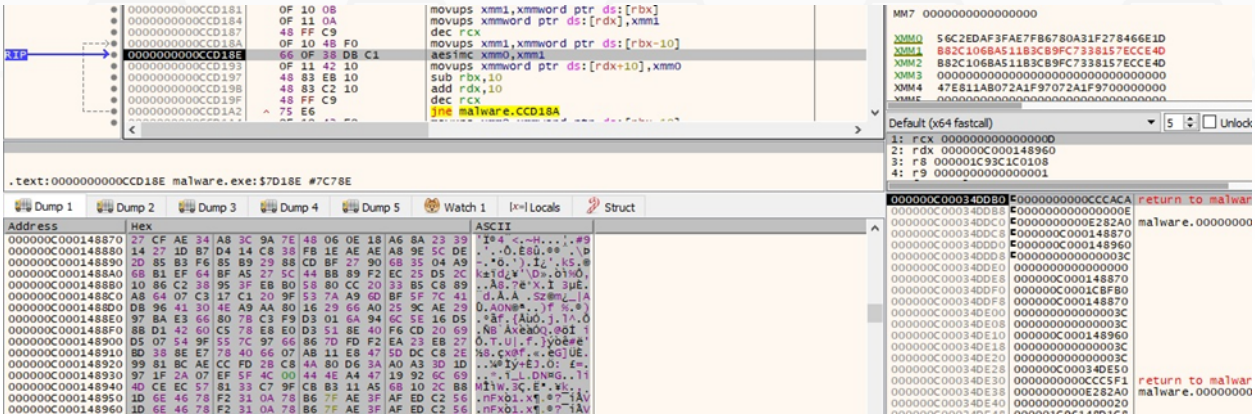

Figure 49

The file is encrypted with 14 rounds of AES encryption (AES256), which is assisted by the aesenc and aesenclast instructions:

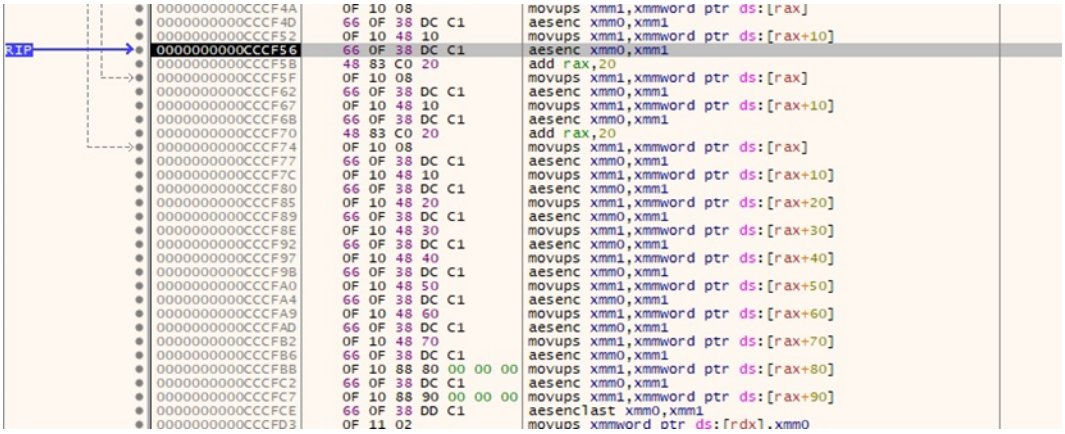

Figure 50

The same key is used to encrypt all files, which is unusual for most ransomware families. The AES key is not encrypted by a public key, which makes the decryption possible if the ransomware encryptor is recovered after its deletion.

The encrypted content is written back to the file using WriteFile:

Figure 51

The malware appends the ".bianlian" extension to all encrypted files (0x1 = **MOVEFILE_REPLACE_EXISTING**):

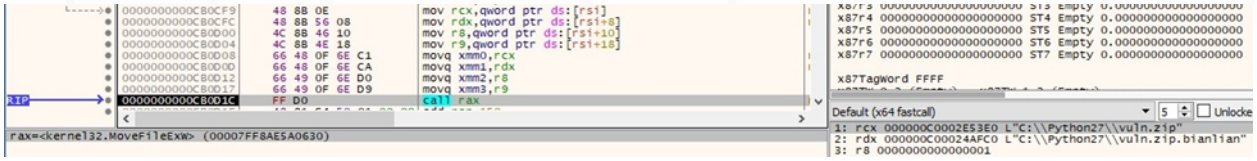

Figure 52

The encryption operation starts at position 0x3d (61), meaning the first few bytes are not encrypted. Also, the encrypted content size is a multiple of 16 bytes (in the case of small files) or 4096 bytes (in the case of files > 1KB).

An example of an encrypted file is shown below:

vuln.zip.bianlian

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000010	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000020	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00000030	41	41	41	41	41	41	41	41	41	41	41	41	41	02	52	DB	AAAAAAAAAAAAA.RÛ
00000040	18	0D	6B	38	3A	42	BE	64	0F	E9	2C	50	3D	B4	25	DF	..k8:B¾d.é,P=´%ß
00000050	E6	78	99	54	05	A8	A8	E4	08	DF	FA	CD	3A	50	8F	BD	æx™T.¨¨ä.ßúÍ:P.½
00000060	56	22	40	96	CC	B0	71	9C	1E	60	39	D2	FB	49	49	F4	V"@-Ì°qœ.`9ÒûIIô
00000070	75	E0	87	9F	5C	1C	B3	46	A2	83	16	72	87	C3	80	10	uà‡Ÿ\.³F¢ƒ.r‡Ã€.

Figure 53

vuln.zip.bianlian

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000FE0	ED	79	88	8A	CF	42	1B	14	5D	64	3A	10	57	7E	C2	9B	íyˆŠÏB..]d:.W~Â›
00000FF0	7D	AA	83	9C	75	54	42	2C	C3	92	AA	20	3C	A1	4A	4C	}ªƒœuTB,Ã’ª <¡JL
00001000	94	51	7E	55	D6	9B	E7	A9	88	AA	C5	7B	9D	84	E8	C3	"Q~UÖ›ç©ˆªÅ{.„èÃ
00001010	78	06	ED	70	AB	BD	09	1F	60	F7	3F	8A	BB	C4	7B	6C	x.íp«½..`÷?Š»Ä{l
00001020	C9	AC	AA	F8	1D	ED	47	06	EC	2D	B1	D5	15	E3	7F	C1	É¬ªø.íG.ì-±Õ.ã.Á
00001030	85	C1	CE	50	A7	AC	A6	61	AA	B6	16	3F	73	41	41	41	…ÁÎP§¬¦aª¶.?sAAA
00001040	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00001050	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00001060	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
00001070	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA

Figure 54

Indicators of Compromise

SHA256

eaf5e26c5e73f3db82cd07ea45e4d244ccb3ec3397ab5263a1a74add7bbcb6e2

BianLian Ransom Note

Look at this instruction.txt

Process spawned

C:\Windows\System32\cmd.exe /c del <Ransomware path>